CLAIMS

I claim:

1. A method of synchronizing graphics commands in a multi-stage graphics system, comprising:

adding draw commands to entries in a draw command list, the draw commands for drawing graphics on a frame;

associating one or more predicate functions with at least some entries in the list, each predicate function satisfiable upon the occurrence of a condition; and

responsive to the satisfaction of the predicate functions associated with each entry, transferring the draw commands in the entry to a next stage in the graphics system.

2. The method of claim 1, further comprising:

loading textures into a texture memory of the graphics system.

3. The method of claim 2, wherein loading textures comprises:

determining whether a texture can be placed in the texture memory according to a linear, first-fit placement algorithm;

if the determination is positive, loading the texture into the texture memory according to the linear, first-fit placement algorithm.

4. The method of claim 2, wherein loading textures comprises:

adding instructions for loading a texture into a texture load list; and

loading textures into the texture memory according to the texture load list when texture memory is available.

5. The method of claim 4, wherein whether texture memory is available is determined according to a linear, first-fit placement algorithm.

6. The method of claim 1, wherein entries in the list are stored in queue, and wherein draw commands in the entries in the list are transferred to a next stage of the graphics system in a first in first out order.

7. The method of claim 2, wherein the loading comprises:

scaling a texture to align the texture with an address boundary in the texture memory.

8. The method of claim 7, wherein the texture is aligned on a 32-bit address boundary.

9. The method of claim 1, wherein draw commands are added to entries in the draw command list by adding a reference to the draw commands stored in memory.

10. The method of claim 1, wherein the draw command list comprises a FIFO queue.

11. The method of claim 1, wherein a draw command is associated with a texture, the method further comprising:

associating a predicate function with the entry in the draw command list that holds the draw command associated with the texture, the predicate function satisfied responsive to the texture's being loaded into a texture memory of the graphics system.

12. The method of claim 1, wherein a draw command is associated with a screen buffer in the graphics system, the method further comprising:

associating a predicate function with the entry in the draw command list holding the drawing command, the predicate function satisfied responsive to the screen buffer's being a back buffer.

13. The method of claim 1, further comprising:

associating a completion function with an entry in the draw command list, the completion functions adapted to execute responsive to the associated entry's being transferred to a next stage of the graphics system.

14. The method of claim 1, wherein the graphics system is adapted to execute an interrupt service routine (ISR) responsive to an occurrence of a predetermined event in the graphics system, the ISR being adapted to:

examine the predicate functions associated with a first entry in the drawing command list; and

responsive to a determination that all of the predicate functions associated with the first entry are satisfied, transferring the drawing commands in the entry to the next stage in the computer graphics system.

15. The method of claim 14, wherein the predetermined event is a swap event.

16. The method of claim 14, wherein the predetermined event is a loading of a texture into a texture memory of the graphics system.

17. The method of claim 14, wherein the ISR is further adapted to:

determine whether a predicate of the first entry in the drawing command list is satisfied; and

responsive to a determination that the predicate is satisfied, indicate that the predicate is satisfied.

18. A method of managing resources in a multi-stage graphics system, comprising:

while the graphics system is drawing a current frame to a display,

processing graphics data associated with a later frame, the graphics data including draw commands associated with one or more textures;

queuing the draw commands for being transferred to graphics hardware, wherein at least some of the draw commands are associated with one or more conditions; and

transferring each draw command to a next stage of the graphics system upon the satisfaction of all of the draw command's conditions.

19. The method of claim 18, wherein at least some of the conditions include whether a swap event occurred.

20. The method of claim 18, wherein at least some of the conditions include whether the textures associated with the draw command have been loaded into a texture memory.

21. The method of claim 18, further comprising loading the textures into a texture memory according to a linear, first-fit placement algorithm.

22. The method of claim 18, further comprising queuing the loading of textures into a texture memory.

23. A computer program product comprising a computer-readable medium having computer program code embodied therein for managing graphics resources, the computer program code comprising:

a graphics module for processing draw commands, the draw commands associated with one or more textures;

a draw entry module adapted to maintain a draw command list of draw entries, each draw entry containing one or more draw commands, wherein at least some draw entries further contain one or more predicate functions satisfiable upon the occurrence of a condition; and

a draw entry logic module adapted to transfer each draw entry's draw commands for processing if all of the draw entry's predicate functions are satisfied.

24. The computer program product of claim 23, further comprising:

a texture loading module adapted to load textures into a texture memory according to a linear, first-fit placement algorithm.

25. The computer program product of claim 23, further comprising:

a texture load list module adapted to add instructions for loading textures to a texture load list; and

a loading list logic module adapted to load textures into the texture memory when texture memory is available according to the texture load list.

26. The computer program product of claim 23, wherein a predicate function is satisfied if a particular texture has been loaded into the texture memory.

27. The computer program product of claim 23, wherein a predicate function is satisfied upon a swap event.

28. The computer program product of claim 23, wherein each draw entry contains a reference to one or more draw commands stored in a memory.

29. A computer program development environment for producing a computer program product, the computer program product comprising:

a graphics module for processing draw commands, the draw commands associated with one or more textures;

a draw entry module adapted to maintain a draw command list of draw entries, each draw entry containing one or more draw commands, wherein at least some draw entries further contain one or more predicate functions satisfiable upon the occurrence of a condition; and

a draw entry logic module adapted to transfer each draw entry's draw commands for processing if all of the draw entry's predicate functions are satisfied.

30. The computer program product of claim 29, further comprising:

a texture loading module adapted to load textures into a texture memory according to a linear, first-fit placement algorithm.

31. The computer program product of claim 29, further comprising:

a texture load list module adapted to add instructions for loading textures into a texture load list; and

a loading list logic module adapted to load textures into the texture memory when texture memory is available according to the texture load list.

32. The computer program product of claim 29, wherein a predicate function is satisfied if a particular texture has been loaded into the texture memory.

33. The computer program product of claim 29, wherein a predicate function is satisfied upon a swap event.

34. The computer program product of claim 29, wherein each draw entry contains a reference to one or more draw commands stored in a memory.

35. A method of developing a computer program product, the method comprising:

receiving graphics data and program code from a developer; and

combining the graphics data and program code with:

a graphics module for processing draw commands, the draw commands associated with the graphics data,

a draw entry module adapted to maintain a list of entries, each entry containing one or more draw commands, wherein at least some entries further contain one or more predicate functions satisfiable upon the occurrence of a condition, and

a list logic module adapted to transfer each entry's draw commands for processing if all of the entry's predicate functions are satisfied;

thereby yielding the computer program product.

36. The method of claim 35, wherein combining further comprises combining the graphics data and program code with:

a texture loading module adapted to load textures into a texture memory according to a linear, first-fit placement algorithm.

37. The method of claim 35, wherein combining further comprises combining the graphics data and program code with:

a texture load list module adapted to add instructions for loading textures into a texture load list; and

a loading list logic module adapted to load textures into the texture memory when texture memory is available according to the texture load list.

38. The method of claim 35, further comprising:

optimizing the computer program product to create a plurality of applications compatible with different hardware platforms.